éduSCOL
Informer et accompagner les professionnels de l'éducation

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE DE L'ÉDUCATION NATIONALE ET DE LA JEUNESSE

VOIE GÉNÉRALE

2DE | 1RE | TLE

Numérique et Sciences informatiques

ENSEIGNEMENT
SPÉCIALITÉ

INTERACTION CLIENT-SERVEUR

1. Requêtes HTTP

1.1. Introduction

HTTP est un protocole de transmission au même titre que FTP.
Il permet à un client de récupérer auprès d'un serveur web des données. Si le client est un navigateur web, ce protocole permet d'obtenir les données nécessaires à l'affichage d'une page internet.
Le protocole fonctionne par l'intermédiaire de requêtes émises par le client et de réponses fournies par le serveur.

1.2. Étude détaillée d'un cas pratique

Examinons la demande d'une page web telle que eduscol.education.fr par un navigateur internet.
Le navigateur commence par demander, à un serveur de noms de domaines (DNS), l'adresse IP de eduscol.education.fr.
Celui-ci lui fournit 185.75.143.28.
La communication par IP n'est pas suffisante car sur un ordinateur peuvent fonctionner diverses applications utilisant internet comme un navigateur ou un logiciel de messagerie. De plus, les paquets de données transitant par le protocole IP ont une taille limite d'environ 1500 octets.
On utilise alors TCP, un protocole de connexion situé sur une couche supérieure à celle d'IP :

- TCP initie et termine les connexions de manière courtoise.
- TCP permet de communiquer au travers de ports réservés : 80 ou 8080 pour HTTP, 21 pour FTP.
- TCP découpe les données en paquets compatibles avec la taille requise pour IP.
- TCP numérote ces paquets et vérifie qu'ils sont bien arrivés à destination au moyen d'accusés de réception délivrés par le destinataire.
- A l'arrivée, grâce aux numéros, TCP permet d'assembler les données.

Le navigateur contacte donc directement le serveur web à l'adresse 185.75.143.28 :80 en utilisant la couche TCP/IP.
La connexion s'établit grâce au 3way-handshake.

- Le client : "Bonjour, je voudrais me connecter au port 80".
- Le serveur : "Bonjour, vous pouvez vous connecter".
- Le client : "Merci, je suis connecté".

Le client envoie alors la requête HTTP au serveur web.

```
GET / HTTP/1.1
Host: eduscol.education.fr
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:67.0) Gecko/20100101
    ↪ Firefox/67.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1

```

La requête possède toujours la même forme :

- ▷ la ligne de requête découpée en *méthode URL Version*. Ici, on utilise la méthode `GET` pour obtenir une ressource. Celle-ci est la racine (/) du site web. Et le protocole HTTP est la version 1.1.
- ▷ les champs de l'en-tête permettant de préciser la demande. L'hôte est `eduscol.education.fr`. Le navigateur web (`User-Agent`) est Firefox. Le langage préféré est le français (`Accept-Language`). Les ressources peuvent être compressées (`gzip`).
- ▷ une ligne vide.
- ▷ le corps de la requête (éventuellement).

Le serveur web renvoie l'entête suivant

```
HTTP/2.0 200 OK
server: nginx
date: Tue, 16 Jul 2019 15:27:37 GMT
content-type: text/html; charset=UTF-8
content-length: 115892
set-cookie: PHPSESSID=s6v6hha07cp56gon9q8apjb494; path=/; HttpOnly
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
cache-control: max-age=3600
age: 2238
x-cache: HIT
x-cache-ttl-remaining: 1361.339
accept-ranges: bytes
x-content-type-options: nosniff
X-Firefox-Spdy: h2

```

suivi des données c'est-à-dire de la page web au format HTML.
La réponse possède, elle aussi, une réponse formatée :

- ▷ la ligne de statut découpée en *Version Code Signification*. Le serveur utilise une version supérieure de HTTP. Le code renvoyé est 200 ce qui signifie que tout s'est bien passé et on a la traduction en OK ;
- ▷ les champs de l'en-tête où l'on apprend le type de serveur, la date, le type MIME de la réponse, sa longueur, etc. ;
- ▷ une ligne vide ;
- ▷ la réponse qui est une page HTML ici.

La connexion TCP/IP est alors fermée.
Le navigateur analyse la page et, pour chaque lien présent, réalise toutes les étapes précédentes avant d'afficher le résultat. Une page web moderne peut nécessiter près de 100 requêtes : la page eduscol.education.fr a nécessité par exemple 126 requêtes.

1.3. Approfondissement de HTTP

L'étude des requêtes et des réponses HTTP peut être réalisée dans Firefox grâce au menu Outils/Développement web/Réseau.
Les méthodes disponibles pour HTTP sont les suivantes : GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE, PATCH.
GET permet d'obtenir une ressource.
PUT, DELETE et PATCH permettent de modifier des données sur le serveur : il est évidemment nécessaire d'être authentifié pour réaliser ces changements et le serveur doit être configuré pour autoriser ces changements.
HEAD permet d'obtenir seulement l'en-tête de la réponse.
POST permet d'envoyer une ressource au serveur : cette méthode est souvent utilisée lors du remplissage d'un formulaire.
Les codes de statuts commencent par 100, 200, 300, 400 ou 500.

- 200 et suivants indiquent une réussite.
- 300 et suivants indiquent un déplacement de la ressource demandée.
- 400 et suivants indiquent une erreur du client : requête mal formulée ou ressource inexistante (la fameuse erreur 404)
- 500 et suivants indiquent une erreur du serveur.

HTTP est sans état, c'est-à-dire qu'il n'y a pas de lien entre deux requêtes réalisées sur la même connexion. Cela pose problème en particulier sur des sites de commerce en ligne avec un panier d'achat que l'on remplit au fur et à mesure. Mais HTTP n'est pas sans session : on ajoute des cookies au flux HTTP, que l'on stocke sur le client, ce qui permet de maintenir une session pour l'utilisateur et donc de remplir le panier d'achat !
HTTP a connu plusieurs révisions :

- Version 1.0 est ce que l'on vient de décrire.
- Version 1.1 : plusieurs requêtes peuvent se succéder au sein de la même connexion TCP/IP (`keep-alive`) mais si une requête n'aboutit pas, elle ralentit les suivantes. Une technique d'optimisation consiste à ouvrir plusieurs connexions TCP/IP en parallèle sur le même site mais les navigateurs modernes limitent le nombre de connexions parallèles à 6.
- Version 2.0 : plusieurs requêtes en parallèle gérées au sein de la même connexion TCP/IP. Possibilité de push (les données sont envoyées au client alors qu'il ne les a pas encore demandées). Compression des en-têtes.

Ces révisions permettent d'accélérer le chargement des pages web qui ont une certaine tendance à l'embonpoint ! De plus, les données sont mises en cache dans le navigateur. Cela permet de ne pas les recharger auprès du serveur lorsque l'on utilise les boutons *Précédent* et *Suivant* du navigateur.

2. Formulaires

2.1. Construction

Sur de nombreuses pages web, des formulaires sont utilisés que ce soit pour une authentification par identifiant et mot de passe ou pour une recherche d'informations.
Le formulaire est un élément codé en HTML

```
1 <form method="get" action="action.php">
2 <label for="ident">Votre identifiant</label> :
3 <input type="text" name="identifiant" id="ident"/>
4 <input type="submit" value="Envoyer" />
5 </form>
```

Un formulaire se trouve contenu dans `<form [...] /form>`.
Dans la balise `<form>`, on ajoute la méthode HTTP utilisée (ici GET) et le lien qui sera activé lors de l'envoi du formulaire (`action.php`).
Ensuite, grâce à `<input[...] />`, on crée deux objets :

- un champ de texte (ligne 3)
- et un bouton pour valider le formulaire (ligne 4).

Le champ de texte est précédé d'un label référencé par `ident`. Cette référence est utilisée dans `input` avec `id` afin qu'un clic sur "Votre identifiant" place le curseur dans le champ de texte correspondant. `identifiant` est une "variable" qui recevra le texte entré dans le champ et qui sera transmise à `action.php`.

2.2. GET ou POST

Lors de l'utilisation de la méthode GET dans un formulaire, les données sont transmises en clair dans la barre d'adresse du navigateur. Dans les faits, on obtient `action.php?identifiant=<texte entré>`.
Cela pose des problèmes de sécurité. En effet, au lieu du type `"text"`, on peut utiliser le type `"password"`. Lors de l'entrée du mot de passe, les lettres sont remplacées par des points. Mais lors de l'envoi des données, le mot de passe apparaît en toutes lettres dans la barre d'adresse !
La méthode POST transmet les données dans le corps de la requête : les données sont donc moins visibles.

2.3. Gestion des données

Pour gérer les données provenant du formulaire, on utilise un script php `action.php`. Le code le plus simple possible est le suivant :

Code 1 – action.php

```
<html>
<body>
<?php
echo $_POST['identifiant'];
?>
</body>
</html>
```

On suppose que la méthode utilisée dans l'envoi du formulaire est POST. Grâce à PHP, on peut alors récupérer le texte et l'afficher. PHP est un langage de script executé au niveau du serveur. Celui-ci doit évidemment accepter de gérer le PHP. Pour tester le formulaire et le script en local, on peut installer WAMP sous Windows, MAMP sous Mac OS X ou LAMP sous Linux. AMP est l'acronyme de Apache (serveur web), MySQL (gestionnaire de base de données), PHP.

3. Sécurité

HTTP transmet le texte brut. Pour que les données soient transmises de manière sécurisée, on utilise HTTPS. Lorsque la communication avec un site WEB est sécurisée, la barre d'adresse commence par un cadenas.
HTTPS est une association de HTTP et de SSL.
SSL (Secure Socket Layer) vient en complément de TCP/IP. Ses avantages sont

- la confidentialité : on ne peut espionner les données. La connexion est chiffrée.
- l'intégrité des données : on ne peut les modifier.
- l'authentification : on est sûr de l'identité du client et du serveur grâce à des certificats présents dans le navigateur, fournis par des sociétés tierces à qui on fait implicitement confiance.

L'évolution récente de SSL est TLS (Transport Layer Security) qui accompagne la version 2 de HTTP.
De plus en plus de sites WEB basculent en HTTPS sous la pression des navigateurs internet. En effet, au lieu d'indiquer simplement qu'un site en HTTPS est sécurisé avec le fameux cadenas, les navigateurs indiquent que les sites HTTP ne sont pas sécurisés.
Il faut bien se rappeler que seule la communication est sécurisée : rien ne nous assure que les données confidentielles fournies au site sont elles-mêmes protégées sur le serveur.

4. Références

- Le descriptif de HTTP fourni par Mozilla : https://developer.mozilla.org/fr/docs/Web/HTTP.
- Les formulaires en HTML : https://developer.mozilla.org/fr/docs/Web/Guide/HTML/Formulaires.
- Les scripts PHP : https://www.w3schools.com/php7/default.asp ou https://openclassrooms.com/fr/courses/918836-concevez-votre-site-web-avec-php-et-mysql.